UNIL | Université de Lausanne

Exercise sheet – OpenMP

Simon Scheidegger
simon.scheidegger@unil.ch
Sept 6th, 2018

Cowles Foundation – Yale

1. Normalize a vector

normalize vec.f90/normalize_vec.cpp:

- Find code snippets here:
  YaleParallel2018/day2/code/supplementary_material_omp
- Write a makefile to compile the file
- Write a parallel version of the subroutine normalize vector(),
  i.e. normalize_vector_omp(v,n) (go to line 37 of the example).
- Scale the vector by the norm to give length=1.
- Experiment with different OMP_NUM_THREAD counts.
- Play with different sizes of the vector and observe running times.
- Run the code by using an adjusted job submission script
  (openmp.sh).

2. Dot product

dot prod.f90/dot_prod.cpp:

- This code finds the dot product of two vectors.
- Add OpenMP directives to parallelize the code.
- Write a makefile to compile the file.
- Experiment with different OMP_NUM_THREAD counts.
- Play with different sizes of the vector and observe running times.
- Run the code by using an adjusted job submission script (openmp2.sh).
- How does it scale from 1 to 8 threads? (submit 1 to 8 thread job).
- Try arrays of different length e.g. 10,000 up to 500,000,000 entries.
- How does it affect scaling? (remember Amdahl's Law?).

3. Estimating Pi with Monte Carlo and OpenMP

Let's consider the problem of estimating Pi by utilizing the Monte Carlo method. Suppose you have a circle inscribed in a square (as in the figure). The experiment simply consists of throwing darts on this figure completely at random (meaning that every point on the dartboard has an equal chance of being hit by the dart). How can we use this experiment to estimate Pi?

The answer lies discovering the relationship between the geometry of the figure and the statistical outcome of throwing the darts.

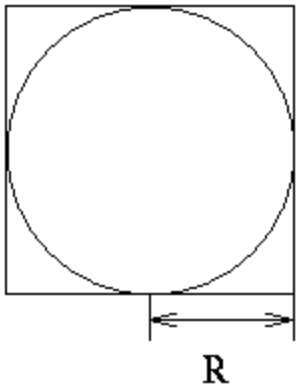
Let's first look at the geometry of the figure. Let's assume the radius of the circle is R, then the Area of the circle = Pi R^2 and the Area of the square = 4 R^2.

Now if we divide the area of the circle by the area of the square we get Pi / 4.

But, how do we estimate Pi by simulation? In the simulation, you keep throwing darts at random onto the dartboard. All of the darts fall within the square, but not all of them fall within the circle. Here's the key. If you throw darts completely at random, this experiment estimate the ratio of the area of the circle to the area of the square, by counting the number of darts in each.

Our study of the geometry tells us this ratio is Pi/4. So, now we can estimate Pi as

Pi = 4 x (Number of Darts in Circle) / (Number of Darts in Square).



→ Revisit this example by adding OpenMP clauses.
→ Experiment with the number of random number you create (N = 100, 1,000, 10,000).
→ Run the code both in batch mode with varying number of threads.

4. Parallel Dynamic Programming

- Look at OpenMP parallel DP code (look at the routine solver.cpp – nested/non-nested OpenMP parallelization).
- **Check the speed-up** for combinations of threads and a variety of **the discretization level** on one node.
- Generate Speed-up and Efficiency graphs (normalize to the 1 CPU result).
- Ensure that the serial and parallel return the same results.